



映画検索サイト

JDBC演習5 レポート 18NC021 カトリスザン



レポートが正常に表示されない場合は最新のブラウザで下記のリンクを開いてください。

<https://movie.sujank.me/assets/report.html>

<https://movie.sujank.me/assets/report.html>



本アプリケーションは下記のリンクで公開されており、クリックするとアクセスできるようになっています。また、下記のリンクをクリック出来ない場合は movie.sujank.me から直接アクセスできる。
2022/08/31までアクセスできることを保証する。

Movie Search

 <https://movie.sujank.me>

本アプリケーションのソースコードや実行方法はGitHubに公開されている。

GitHub - falcon78/movie_search: TMDBのデータベースダンプを使用して作られた映画検索サイト

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

 https://github.com/falcon78/movie_search

falcon78/
movie_search

TMDBのデータベースダンプを使用して作られた映画
検索サイト

👤 1 Contributor 📄 0 Issues ⭐ 0 Stars 🍴 0 Forks

1. 概要

本アプリケーションは映画の名前・ジャンル・映画を制作した会社の会社名で映画を検索できる映画検索ウェブアプリケーションである。

このアプリはホームページ・検索結果一覧ページ・映画詳細ページの三つのページに分かれてる。ホームページから検索したいテキストを入力して検索ボタンを押下すると検索結果の一覧が表示される。そして検索結果の一覧から映画のタイトルをクリックするとその映画の詳細ページに遷移される。

その様子を下記に示す。

ホームページ

Movie Search

Movie

top gun

Search

検索結果一覧画面

一覧ページでは各映画のタイトル・公開日・上映時間・レビュー件数・レビュースコア・公開状態・総利益 (USドル)・人気度の情報を表示している。

下記に映画を検索した時の検索結果の画面のスクリーンショットを掲載する。

- 映画のタイトルで検索

Movie Search

Movie

top gun

Search

Title	Release Date	Runtime (m)	Rating Count	Rating	Status	Revenue	Popularity
Top Gun	1986	110	1,736	6.7	Released	356,830,601	20

Previous1 / 1Next

- 映画のジャンルで検索

Genre

Animation

Search

Title	Release Date	Runtime (m)	Rating Count	Rating	Status	Revenue	Populatiry
Toy Story	1995	81	5,415	7.7	Released	373,554,033	22
Baltq	1995	78	423	7.1	Released	11,348,324	12
Pocahontas	1995	81	1,509	6.7	Released	346,079,773	13
A Goofy Movie	1995	78	404	6.7	Released	35,348,597	10
Gumby: The Movie	1995	77	2	5	Released	0	-
The Swan Princess	1994	89	251	6.5	Released	9,771,658	9
The Lion King	1994	89	5,520	8	Released	788,241,776	22
The Secret Adventures of Tom Thumb	1993	61	8	7.1	Released	0	-
The Nightmare Before Christmas	1993	76	2,135	7.6	Released	75,634,409	18
The Pagemaster	1994	80	178	6.2	Released	13,670,688	7
Aladdin	1992	90	3,495	7.4	Released	504,050,219	16
Snow White and the Seven Dwarfs	1937	83	1,973	6.9	Released	184,925,486	16
Beauty and the Beast	1991	84	3,029	7.5	Released	377,350,553	23
Pinocchio	1940	88	1,412	6.9	Released	84,300,000	14
Heavy Metal	1981	90	150	6.3	Released	0	8
The Aristocats	1970	78	1,287	7.1	Released	55,675,257	10
All Dogs Go to Heaven 2	1996	82	50	5	Released	8,620,678	5
James and the Giant Peach	1996	79	375	6.2	Released	28,921,264	13
Space Jam	1996	88	1,335	6.5	Released	250,200,000	11
Oliver & Company	1988	74	372	6.5	Released	74,151,346	11

Previous

1 / 331

Next

- 映画制作会社の会社名で検索



Production



Pixar

Search

Title	Release Date	Runtime (m)	Rating Count	Rating	Status	Revenue	Populatiry
Toy Story	1995	81	5,415	7.7	Released	373,554,033	22
A Bug's Life	1998	95	2,379	6.8	Released	363,258,859	17
Toy Story 2	1999	92	3,914	7.3	Released	497,366,869	18
Monsters, Inc.	2001	92	6,150	7.5	Released	562,816,256	26
Finding Nemo	2003	100	6,292	7.6	Released	940,335,536	25
The Incredibles	2004	115	5,290	7.4	Released	631,442,092	22
Luxo Jr.	1986	2	148	7.1	Released	0	7
Cars	2006	117	3,991	6.6	Released	461,983,149	19
Ratatouille	2007	111	4,510	7.5	Released	623,722,818	21
WALL-E	2008	98	6,439	7.8	Released	521,311,860	16
Up	2009	96	7,048	7.8	Released	735,099,082	19
Partly Cloudy	2009	5	335	7.9	Released	0	13
Toy Story 3	2010	103	4,710	7.6	Released	1,066,969,703	17
Day & Night	2010	6	272	7.6	Released	0	6
BURN-E	2008	8	250	7.8	Released	0	8
Cars 2	2011	106	2,088	5.8	Released	559,852,396	14
Mike's New Car	2002	4	141	6.9	Released	0	7
Lifted	2006	5	232	7.6	Released	0	6
Brave	2012	93	4,760	6.7	Released	538,983,207	16
Presto	2008	5	371	8	Released	0	9

Previous


1 / 6

Next

映画詳細画面

一覧ページで映画のタイトルがリンクになっており、クリックすると下記のように映画の詳細ページが表示される。詳細ページでは映画のポスター・あらすじ・評価・公開日・ジャンル一覧・制作会社一覧を表示する。





Interstellar

The adventures of a group of explorers who make use of a newly discovered wormhole to surpass the limitations on human space travel and conquer the vast distances involved in an interstellar voyage.

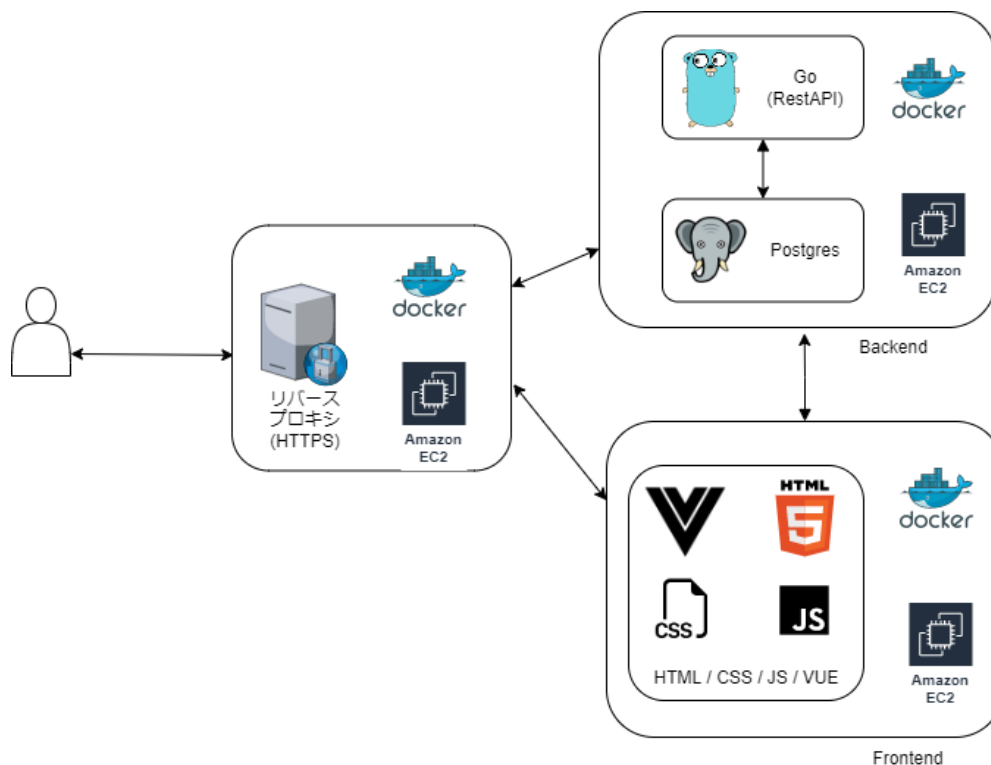
Rating: 8.371/10 (28869)

Released on 2014-11-05

Genres
Adventure
Drama
Science Fiction

Productions
Paramount Pictures
Legendary Pictures
Warner Bros.
Syncopy
Lynda Obst Productions

2. 設計方針



システムの構成は上記の図のようになっている。

Dockerを使うことでアプリケーションの環境構築やデプロイを簡易にできるようにしている。

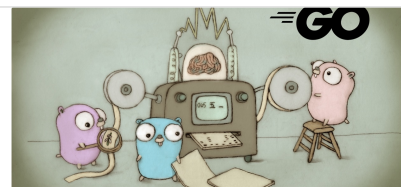
アプリケーションはAWSのEC2インスタンスで動いている。また、サーバとユーザー間はCaddyとLet's Encryptを用いてHTTPS通信を実現している。

また、アプリケーションの実行方法は [README.md](#) ファイルに記載している。

The Go Programming Language

Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.

<https://go.dev/>



Vue.js - The Progressive JavaScript Framework | Vue.js

An approachable, performant and versatile framework for building web user interfaces. Builds on top of standard HTML, CSS and JavaScript with intuitive API and world-class documentation. Truly reactive, compiler-optimized rendering system that rarely requires manual optimization. A

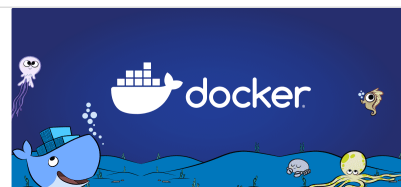
<https://vuejs.org/>



Home - Docker


Learn how Atomist will help Docker meet the challenge of securing secure software supply chains for development teams.

<https://www.docker.com/>



Caddy 2 - The Ultimate Server with Automatic HTTPS

Caddy is both a flexible, efficient static file server and a powerful, scalable reverse proxy. Use it to serve your static site with compression, template evaluation, Markdown rendering, and more. Or use it as a dynamic reverse proxy to any number of backends, complete with active and passive

 <https://caddyserver.com/>



Secure and resizable cloud compute - Amazon EC2 - Amazon Web Services

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 500 instances and choice of the latest processor, storage, networking, operating system, and purchase model to help you best match the needs of your workload.

 <https://aws.amazon.com/ec2/>



3. プログラムの説明

データベース

映画のデータはTMDBのデータダンプから必要なものを抽出し、データベースに追加している。

テーブルの構成は下記に示す。

```
create table genres
(
    id int not null primary key,
    name varchar not null
);

create table productions
(
    id int not null primary key,
    name varchar not null
);

create table movies
(
    id serial primary key,
    adult bool not null,
    budget bigint,
    homepage varchar,
    imdb_id varchar,
    overview varchar,
    popularity float,
    poster varchar,
    release_date timestamp,
    revenue bigint,
    runtime float,
    status varchar,
    title varchar,
    vote_average float,
    vote_count int
);

create table movie_genres
(
    id serial primary key,
    movie_id int references movies (id) not null,
    genre_id int references genres (id) not null
);

create table movie_productions
(
    id serial primary key,
    movie_id int references movies (id) not null,
    production_id int references productions (id) not null
);

create index title on movies (title);
create index genre_name on genres (name);
create index production_name on genres (name);
```

```

create view movie_genre_view as
select movie_id as id,
       genre_id,
       name,
       adult,
       budget,
       homepage,
       imdb_id,
       overview,
       popularity,
       poster,
       release_date,
       revenue,
       runtime,
       status,
       title,
       vote_average,
       vote_count
from movie_genres
     join genres g on movie_genres.genre_id = g.id
     join movies m on movie_genres.movie_id = m.id;

create view movie_production_view as
select movie_id as id,
       production_id,
       adult,
       budget,
       homepage,
       imdb_id,
       overview,
       popularity,
       poster,
       release_date,
       revenue,
       runtime,
       status,
       title,
       vote_average,
       vote_count,
       name
from movie_productions
     join movies m on movie_productions.movie_id = m.id
     join productions p on movie_productions.production_id = p.id;

```

TMDBデータダンプのCSVのJSON形式が間違っていたため、Javascriptでデータ形式を修正している。そして修正したデータをGoでインポートし、データベースに追加している。

データダンプの形式を修正するプログラムは `script.js` の中に入っている。

また、データをデータベースに追加するコードの一部を下記に示す。

```

if movieInfos, productions, genres, err := utils.InsertCsvToDatabase(CsvFile); err != nil {
    panic(err)
} else {
    repo := repository.NewRepository(db)
    if err := repo.InsertAllGenres(genres); err != nil {
        panic(err)
    }
    if err := repo.InsertAllProductions(productions); err != nil {
        panic(err)
    }
    if err := repo.InsertMovieInfo(movieInfos); err != nil {
        panic(err)
    }
}
}

```

アプリケーション

本アプリケーションはフロントエンドとバックエンドに分かれている。フロントエンド側はRestAPIを使ってバックエンドサービスと通信する。

フロントエンドはHTML・CSS・Javascript (Vue.js) を用いて実装されている。Vueを使うことによりHTMLの生成がすべてクライアントサイドで行われ、サーバサイドでHTMLを生成しないためサーバに負荷がかからないようにしている。

バックエンドのRestAPIの仕様とそれらを実装するコードの一部は下記のようにになっている。

- 検索ルート

- `GET /api/search/movies?searchBy=<string>&searchText=<string>&page=<int>&pageCount=<int>`

```
func (a *app) searchMovies(c echo.Context) error {  
    // 省略  
}
```

- 映画の詳細情報取得ルート

- `GET /api/movie/<movieId>`

```
func (a *app) GetMovieInfo(c echo.Context) error {  
    // 省略  
}
```

- 映画のジャンルと制作会社の情報取得ルート

- `GET /api/movie/<movieId>/additionalInfo`

```
func (a *app) GetMovieAdditionalInfo(c echo.Context) error {  
    // 省略  
}
```

4. 実行結果

映画のタイトルで検索した時の実行結果

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f73460d9-329e-4793-8daf-0147483563db/Movie_Search_-_Google_Chrome_2022-07-19_22-41-11.mp4

ジャンルで検索した時の実行結果

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/90eb8d05-88e2-44ec-b561-9175e83fef23/Movie_Search_-_Google_Chrome_2022-07-19_22-43-31.mp4

制作会社名で検索した時の実行結果

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d147d15a-2b62-4b6b-881e-b7f4cda53b7a/Movie_Search_-_Google_Chrome_2022-07-19_22-45-01.mp4

考察

本アプリケーションは時間の都合上実装できていない機能が結構ある。例えばユーザーがレビューを投稿できる機能や映画をお気に入りリストに入れる機能、またその映画を閲覧できる配信サービスへのリンクを表示する機能などを実装したい。

また、バックエンドでもデプロイにKubernetesを使ったり、負荷分散対策するなどをしたい。